Transcrição

Nesta etapa, abordaremos as **Medidas Separatrizes**.

Anteriormente, mencionamos este assunto quando falamos sobre a mediana que divide uma variável em duas partes iguais.

A partir de agora, aprenderemos sobre as seguintes medidas:

- **Quartis**: divide a variável em **quatro** partes iguais quanto ao número de elementos de cada uma;
- **Decis**: divide em **dez** partes iguais;
- **Percentis**: divide em **cem** partes iguais.

Estas medidas são interessantes porque **não são influenciadas** por valores extremos de uma distribuição, diferente da média em relação à `Renda`, c central da distribuição desses dados.

Estes cálculos permitem análises importantes a partir de um **ponto de referência**, como o salário mínimo por exemplo. Também poderemos constru

Na parte "4.1 Quartis, decis e percentis" dentro da nova seção, aprenderemos como obter esses resultados.

Quando queremos dividir uma série em "n" partes, precisaremos de "**n -1**" divisores. No caso da mediana por exemplo, precisaremos de somente um

O mesmo acontecerá com as demais medidas quartis, decis e percentis. Na primeira célula, pegaremos os dados da `Renda` e aplicaremos o já conhe

```
dados.Renda.quantile()
```

Executando este método mesmo sem o parâmetro com valor padrão `q = 0.5`, obteremos o mesmo resultado `1200.0` da mediana.

Também poderemos passar uma lista com as divisões da medida; por exemplo, se queremos calcular os quartis, precisaremos de três valores; o prim

Dentro de `quantile()`, passaremos `0.25` dentro dos colchetes como primeiro divisor. O seguinte é justamente a mediana que divide a variável ao

Por fim, o terceiro quartil será o contrário do primeiro, sendo os 75% primeiros valores abaixo, e os 25% acima. portanto, passaremos `0.75` como ú

```
dados.Renda.quantile([0.25, 0.5, 0.75])
```

```
0.25    788.0
0.50   1200.0
0.75   2000.0
Name: Renda, dtype: float64
```

Rodando o código, veremos os valores que dividem cada quartil.

Aprenderemos a realizar a mesma operação para as medidas decis e percentis; aplicaremos a técnica de ***List comprehension***, ou seja, a construção d

Entre colchetes, passaremos `i` para representarmos a variável. Em seguida, escreveremos o `for` e diremos que este `i` varia em um intervalo `ran`

```
[i for i in range(1, 10)]
```

O resultado será uma lista de `1` até `9`, pois para dividir em 10 partes precisamos calcular "10 - 1" para o número correto de divisores, como já sabe

Como queremos apresentar valores decimais, dividiremos a variável `i` por `10`.

```
[i / 10 for i in range(1, 10)]
```

Com isso, exibiremos a lista de `0.1` até `0.9`.

Para calcularmos os decis, bastará reescrevermos a linha de `quantile()` e passarmos o código que construiu esta última lista.

```
dados.Renda.quantile([i / 10 for i in range(1, 10)])
```

```
0.1    350.0
0.2    788.0
0.3    800.0
0.4   1000.0
0.5   1200.0
0.6   1500.0
0.7   1900.0
0.8   2500.0
0.9   4000.0
Name: Renda, dtype:float64
```

Como resultado, veremos todos os decis criados, bem como as porcentagens e valores que dividem.

Na célula seguinte, calcularemos os percentis da mesma maneira, apenas fazendo a divisão de `i` por `100` e estabelecendo o `range()` de `1` a `99`

```
dados.Renda.quantile([i / 100 for i in range(1, 100)])
```

Isso nos retornará uma lista com 99 divisores e seus valores.

Notaremos a repetição do valor `788.0` em muitas posições, o qual dividia os primeiros 20% no cálculos dos decis anteriormente.

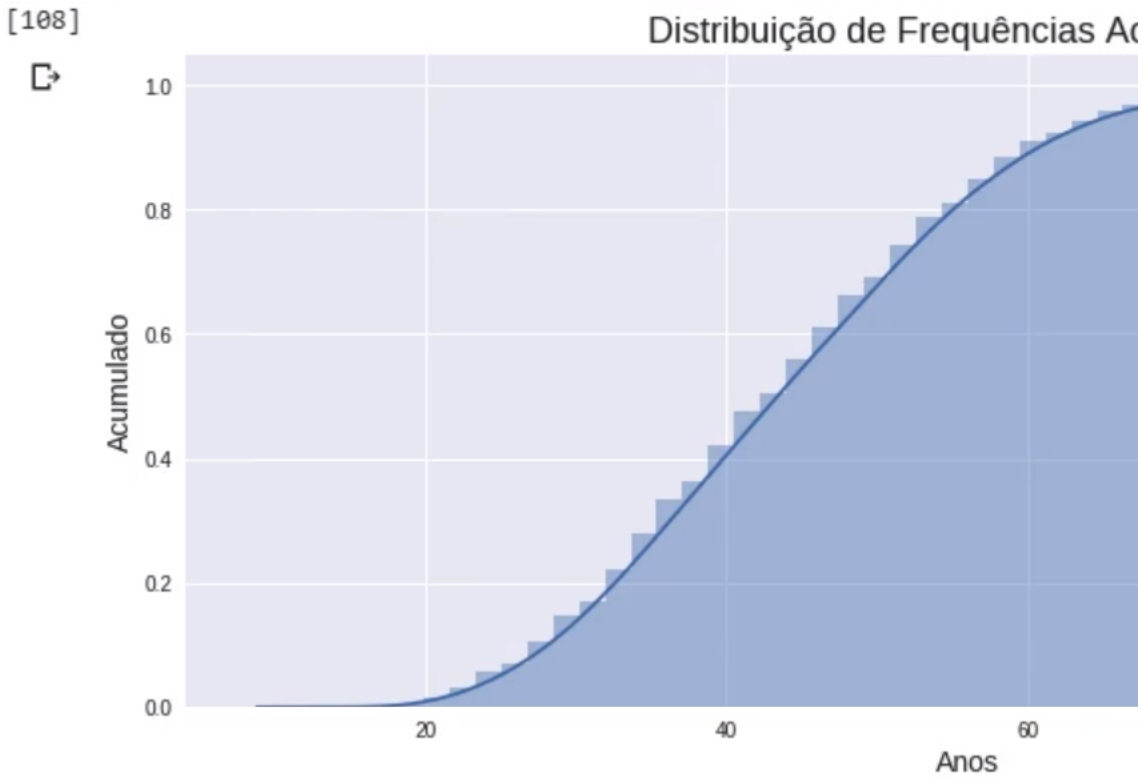
Como o percentil varia a cada 1%, poderemos ver a passagem do valor `788.0` para `789.0` da posição `0.28` para `0.29`, comprovando a maior pr recebem uma renda mensal de até um salário mínimo apenas.

Na célula seguinte, faremos uma representação gráfica `ax` da variável `Idade` utilizando o método `distplot()` da biblioteca Seaborn que herda d

Construiremos um histograma acumulado com dois parâmetros extras; o `hist_kws` sendo igual ao dicionário `'cumulative'` como `True` e a funçã

Em seguida, daremos um título e nomearemos os `ylabel` e `xlabel` como `'Acumulado'` e `'Anos'` com tamanho de fonte `14` respectivamente, e

```
ax = sns.distplot(dados.Idade,
                  hist_kws = {'cumulative': True},
                  kde_kws = {'cumulative': True}),
ax.figure.set_size_inches(14, 6)
ax.set_title('Distribuição de Frequências Acumulada', fontsize=18)
ax.set_ylabel('Acumulado', fontsize=14)
ax.set_xlabel('Anos', fontsize=14)
ax
```



Como resultado, veremos um gráfico acumulativo.

Para termos ainda mais clareza, construiremos a medida decil da `Idade` com o mesmo comando feito para a `Renda` anteriormente.

```
dados.Idade.quantile([i / 10 for i in range(1, 10)])
```

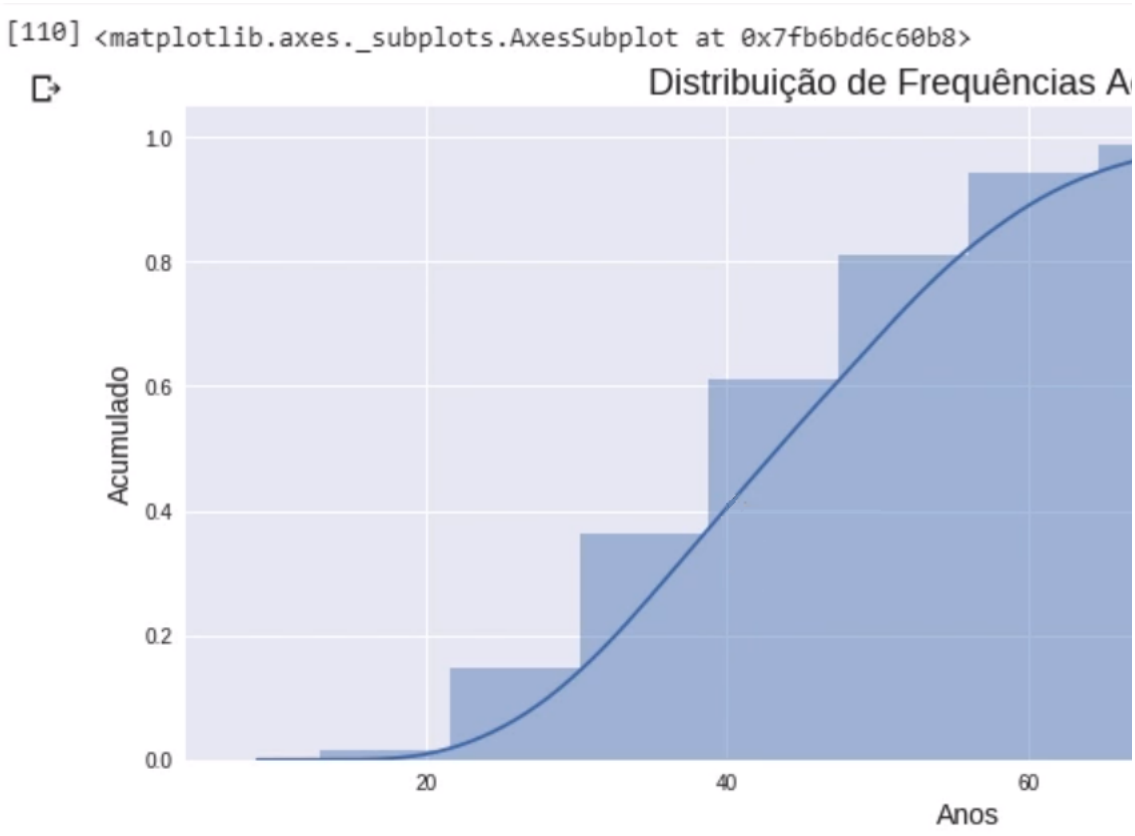
```
0.1   28.0
0.2   33.0
0.3   36.0
0.4   40.0
0.5   43.0
0.6   47.0
0.7   51.0
0.8   55.0
0.9   61.0
Name: Idade, dtype:float64
```

Com o retorno, poderemos constatar que 40% das pessoas entrevistadas possuem até quarenta anos de idade, ou que 90% possuem menos de 61 anos

Em seguida, construiremos sua representação gráfica, passando o número de `bins` como `10` no terceiro parâmetro de `distplot()`

```
ax = sns.distplot(dados.Idade,
                  hist_kws = {'cumulative': True},
                  kde_kws = {'cumulative': True},
                  bins = 10)
ax.figure.set_size_inches(14, 6)
ax.set_title('Distribuição de Frequências Acumulada', fontsize=18)
ax.set_ylabel('Acumulado', fontsize=14)
ax.set_xlabel('Anos', fontsize=14)
ax
```

```
[110] <matplotlib.axes._subplots.AxesSubplot at 0x7fb6bd6c60b8>
```



Com este gráfico mais preciso, poderemos comparar os resultados da tabela de decis e tirar conclusões analíticas.

Observação: A função distplot() está depreciada e devemos usar displot() ou histplot()em seu lugar. Por isso, te aconselhamos uma leitura da bibliot

- Documentação histplot
- Documentação displot

A seguir, veremos a criação do `boxplot` por meio dos quartis calculados na primeira célula deste passo.